



# UNITED STATES PATENT AND TRADEMARK OFFICE

A

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/996,126	11/27/2001	Paul A. Lovvik	SUN-P6876-PIP	1464

22835 7590 11/18/2005

A. RICHARD PARK, REG. NO. 41241  
PARK, VAUGHAN & FLEMING LLP  
2820 FIFTH STREET  
DAVIS, CA 95616

EXAMINER

CHOW, CHIH CHING

ART UNIT PAPER NUMBER

2192

DATE MAILED: 11/18/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	<b>Application No.</b>	<b>Applicant(s)</b>	
	09/996,126	LOVVIK ET AL.	
	<b>Examiner</b>	<b>Art Unit</b>	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) ☒ Responsive to communication(s) filed on 06 September 2005.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) ☒ Claim(s) 1-8, 10-17 and 19-26 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-8, 10-17 and 19-26 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 27 November 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |   |   |
|---|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)             | 4) <input type="checkbox"/> Interview Summary (PTO-413)                     |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)    | Paper No(s)/Mail Date. _____  |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| Paper No(s)/Mail Date _____   | 6) <input type="checkbox"/> Other: _____                                    |

### **DETAILED ACTION**

1. This action is responsive to amendment dated September 02, 2005.
2. Per Applicants' request, independent claims 1, 10, and 19 have been amended, claims 9, 18, and 27 canceled.
3. Claims 1-8, 10-17, 19-26 remain pending.
4. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 09/02/2005 has been entered.

### **Response to Amendment**

5. Applicants' amendment for Claims 1, 10, and 19 have been fully considered respectfully by the examiner but they are not persuasive.
6. The Examiner is maintaining the 35 USC § 102(e) and the 35 USC § 103 Rejections. For the Applicants' convenience they are listed as following, with the amendments requested by the Applicants.

### **Response to Arguments**

7. Applicants' arguments for "the present invention provides a cache within the class analyzer that is not part of the memory system and which stores only the list of dependent classes including the name of the analyzed class." ... "There is nothing within Gerard, either explicit or implicit, which suggests providing a cache within the class analyzer that is not part of the memory system and which stores only the list of dependent classes

Art Unit: 2192

including the name of the analyzed class.” (REMARKS page 9, last paragraph) have been fully considered respectfully by the examiner but they are not persuasive.

Examiner's Response: Either “class analyzer” nor “cache within a class analyzer” is mentioned in any of the independent claims 1, 10, or 19. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., “class analyzer”, “cache within a class analyzer”) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

### **Claim Rejections - 35 USC § 102**

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9. Claims 1, 3-5, 10, 12-14, 19, 21-23 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,442,753 by Scott Neal Gerard et al. (hereinafter “Gerard”).

### CLAIM

1. A method for determining a class dependency that identifies a supporting class on which a target class depends, wherein the target class is defined in an object-oriented programming language, comprising:

- a. receiving a representation of the target class at a first platform-independent virtual machine;
- b. creating a model of the target class from the representation;
- c. analyzing the model to detect references to the supporting class;
- d. if a supporting class is detected, determining a class dependency for the supporting class;
- e. creating a list of dependent classes for the target class and supporting classes;
- f. sharing the list of dependent classes with a second platform-independent virtual machine so that the second platform-independent virtual machine does not need to create the list of dependent classes; and
- g. saving the list of dependent classes of the target class as well as the list of dependent classes of the supporting classes in cache to facilitate subsequent lookups of dependent classes of the target class.

### Gerard

Gerard's Abstract, "A **dependency checking apparatus and method** allows checking the version of classes in an **object-oriented program** to assure the proper version is being used for each release of the software." See Gerard column 2, lines 66-67, "According to a first preferred embodiment, **classes themselves** include static code that **checks dependencies** when the class is loaded." For item a and b, see Gerard column 7, lines 61-62, "begins by **determining the dependencies among classes** (step 410)" – in order to do the analysis, a 'target class' should have been received first. For items b and c, Gerard column 7 lined 63-67, "Once the dependencies have been determined, they are checked (step 420). **If all of the dependencies are correct** (step 430=YES), the **object-oriented program** may be executed (step 440). In the alternative, if one or more dependency is not satisfied (step 430=NO), method 400 returns an error (step 450), which causes the object-oriented program to not be run (*analyzing the model to detect references to the supporting class*)." Also see Gerard's FIG. 4, a 'model' to detect dependencies is disclosed. For items d and e, see Gerard's FIG. 7, the '**Dependor Class**' is same as the target class in current application, and the '**Dependee class**' is same as the supporting class, and the '**List of Dependee Classes**' is the list for supporting classes, one skilled in the art could easily create a list of dependent classes for both the target class and supporting classes. For item e, see Gerard's column 9, lines 19-22, "For example, a file may be used to store dependency

information. Referring to FIG. 7, a file 700 that contains dependency information specifies the dependor class, the version of the dependor class, a list of dependee classes, and a list of valid versions for each dependee class.” For item f, see Gerard’s column 5, lines 54-65, “computer system 100 uses well known virtual addressing mechanisms that allow the programs of computer system 100 to behave as if they only have access to a large, single storage entity, instead of access to multiple, smaller storage entities (e.g., main memory 120 and DASD device 180). Therefore, while certain elements are shown to reside in main memory 120, those skilled in the art will recognize that these are not necessarily all completely contained in main memory 120 at the same time. It should be noted that the term ‘memory’ is used herein to generically refer to the entire virtual memory of computer system 100” – Gerard’s disclosure implies that the dependency information are shared among other platform-independent machines. For item g, Cache memory is also mentioned in Gerard’s disclosure, see column 5, line 5, “Examples of possible additions include: a computer monitor, a keyboard, a **cache memory**, and peripheral devices such as printers.”

3. The method of claim 1, further comprising saving the list of dependent classes to a storage structure.

For the feature of claim 1 see claim 1 rejection. See FIG. 7, diagram showing information stored in an external file (*storage structure*) in accordance with the second embodiment.

4. The method of claim 3, wherein the storage structure is one of a hash table and a database.

For the feature of claim 3 see claim 3 rejection. See claim 3 rejection, “an external file” can be a database. It is

anticipated for those skilled in the art to implement the storage structure using a hash table.

5. The method of claim 1, wherein creating the list of dependent classes includes creating one of a distribution list and a distribution file.

For the feature of claim 1 see claim 1 rejection. See FIG. 1, item 170, Gerard's disclosure is able to distribute through the internet, also see Gerard column 6, lines 30-33, "those skilled in the art will appreciate that the present invention is **capable of being distributed as a program product** in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution."

10. A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for determining a class dependency that identifies a supporting class on which a target class depends, wherein the target class is defined in an object-oriented programming language, the method comprising:

Same as Claim 1 rejection. Gerard's disclosure includes a computer-readable storage medium.

- a. receiving a representation of the target class at a first platform-independent virtual machine;
- b. creating a model of the target class from the representation;
- c. analyzing the model to detect references to the supporting class;
- d. if a supporting class is detected, determining a class dependency for the supporting class;
- e. creating a list of dependent classes for the target class and supporting classes;
- f. sharing the list of dependent classes with a second platform-independent virtual

machine so that the second platform-independent virtual machine does not need to create the list of dependent classes; and  
g. saving the list of dependent classes of the target class as well as the list of dependent classes of the supporting classes in cache to facilitate subsequent lookups of dependent classes of the target class.

12. The computer-readable storage medium of claim 10, wherein the method further comprises saving the list of dependent classes to a storage structure.

For the feature of claim 10 see claim 10 rejection. For the rest of the claim 12 feature see claim 3 rejection.

13. The computer-readable storage medium of claim 12, wherein the storage structure is one of a hash table and a database.

For the feature of claim 12 see claim 12 rejection. For the rest of the claim 13 feature see claim 4 rejection.

14. The computer-readable storage medium of claim 10, wherein creating the list of dependent classes includes creating one of a distribution list and a distribution file.

For the feature of claim 10 see claim 10 rejection. For the rest of the claim 14 feature see claim 5 rejection.

19. An apparatus that determines a class dependency that identifies a supporting class on which a target class depends, wherein the target class is defined in an object-oriented programming language, comprising:

Same as claim 1 rejection. In order for Gearard's disclosure to work, it has to have 'mechanisms' to implemented all the recited steps.

- a receiving mechanism that is configured to receive a representation of the target class at a first platform-independent virtual machine;

- a modeling mechanism that is configured to create a model of the target class from the representation;

- an analysis mechanism that is configured to analyze the model to detect references to the supporting class;

- a supporting mechanism that is configured to determine a class dependency



for the supporting class;

a listing mechanism that is configured to create a list of dependent classes for the target class and supporting classes;

a sharing mechanism that is configured to share the list of dependent classes with a second platform-independent virtual machine so that the second platform-independent virtual machine does not need to create the list of dependent classes; and

a saving mechanism configured to save the list of dependent classes of the target class as well as the list of dependent classes of the supporting classes in cache to facilitate subsequent lookups of dependent classes of the target class.

21. The apparatus of claim 19, wherein the listing mechanism is configured to save the list of dependent classes to a storage structure.

For the feature of claim 19 see claim 19 rejection. For rest of the claim 21 feature see claim 3 rejection.

22. The apparatus of claim 21, wherein the storage structure is one of a hash table and a database.

For the feature of claim 21 see claim 21 rejection. For rest of the claim 22 feature see claim 4 rejection.

23. The apparatus of claim 19, wherein the listing mechanism is configured to create the list of dependent classes, including creating one of a distribution list and a distribution file.

For the feature of claim 19 see claim 19 rejection. For rest of the claim 23 feature see claim 5 rejection.

### **Claim Rejections - 35 USC § 103**

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

Art Unit: 2192

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

11. Claims 2, 6-8, 11, 15-17, 20, 24-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,442,753 by Scott Neal Gerard et al. (hereinafter "Gerard"), in view of U.S. Patent No. 5,787,275 by Shih-Gong Li. (hereinafter "Li").

### CLAIM

2. The method of claim 1, further comprising, identifying classes that an object depends upon by:

- a. receiving a representation of the object;
- b. serializing the referenced object;
- c. parsing the data resulting from the object serialization to identify classes referenced from the target object's properties, configuration, or state; and
- d. determining the dependent classes of the referenced object.

### Gerard / Li

For the feature of claim 1 see claim 1 rejection. Gerard column 4, lines 2-10, "Each object is an identifiable, encapsulated piece of code and data that provides one or more services when requested by a client. Conceptually, an object has two parts, an external object interface and internal object implementation (*representation of the object*). In particular, all object implementation functions are encapsulated by the object interface such that other objects must communicate with that object through its object interface. The only way to retrieve, process or otherwise operate on the object is through the methods defined on the object." Gerard teaches all aspects of claim 2, but he does not mention 'serializing the referenced object', 'parsing the data' and 'determining the dependent classes of the referenced object' specifically, however, Li teaches these in an analogous prior art. For item a, see Li, column 5, lines 27 to column 6, line 43, the 'Inheritance Relationship', 'Usage Relationship' and Implementation Usage Relationship' all used 'object' as the analysis target; see column 5, line 46, "describes that Class Window uses an object of Class DeviceContext as the return of GetDevContext method." And column 6, lines 33-34, "Implementation Usage captures the usage relationships that occurred only in the implementation body

of the **object** rather than in the method's interface". For item b, see Li column 5, lines 19-20, "The relationships retrieved by the **parser** and **stored in the relation data library** (*serializing*) are preferably described in a predicate form". For items c and d, see Li column 2, lines 35-37, "First, an object oriented program in an object oriented source code language is **parsed** for the **immediate class relationship data**. (*determining the dependent classes*) The immediate class relationship data is stored in a relation data library."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Gerard's disclosure of the class dependencies analysis by using object analysis taught by Li, for the purpose of identifying and analyzing class relationships in a object oriented system (Li, column 2, lines 23-25).

6. The method of claim 2, further comprising:  
a. inserting the object into an object database;  
b. determining if the target class and supporting classes for the target class are in the class path; and  
c. adding the target class and supporting classes for the target class to the class path if necessary.

For the feature of claim 2 see claim 2 rejection. For item a, the object is inserted into an object database (*relation data library*). For items b and c, the class analysis and adding the target class to the class path are performed in Gerard (see claim 1 rejection and Gerard FIG. 4).

7. The method of claim 2, further comprising:  
a. retrieving the object from the object database;  
b. determining if the target class and supporting classes for the target class are in the class path; and

For the feature of claim 2 see claim 2 rejection. For item a, see Li, column 10, lines 23-28, "If there are more levels to output, in step 123, the depend level relationship data is **retrieved** from the facts data library for (A,1,L) where (A,1,L) is the argument list for depend (A,1,L) and

c. adding the target class and supporting classes for the target class to the class path if necessary.

A=classname, 1=level number and L=List of dependency data. Next, in step 125, a test is performed to determine whether L=0 which means that the class does not have a dependency list." Li's disclosure teaches 'adding' or retrieving the object from the object database. For items b and c, the class analysis and adding the target class to the class path are performed in Gerard (see claim 1 rejection and Gerard FIG. 4).

8. The method of claim 1, further comprising filtering the list of identified classes to remove duplicate and core class references.

For the feature of claim 1 see claim 1 rejection. Gerard teaches all aspects of claim 2, but he does not mention 'filtering the list of identified classes' specifically, however, Li teaches it in an analogous prior art. See Li column 10, lines 11-16, "In step 113, a test is performed to determine whether there is more than one entry in the Processed.sub.-- Table for class A. If so, it means that the class has already been processed in the session and that there is no need to repeat the following steps (filtering the duplication)."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Gerard's disclosure of the class dependencies analysis by filtering the duplicated classes taught by Li, for the purpose of saving the storage space.

11. The computer-readable storage medium of claim 10, wherein the method further comprises, identifying classes that an object depends upon by:

- a. receiving a representation of the object;
- b. serializing the referenced object;
- c. parsing the data resulting from the object serialization to identify classes referenced from the target object's

For the feature of claim 10 see claim 10 rejection. For the rest of the claim 11 feature see claim 2 rejection.

properties, configuration, or state; and

d. if a target class is identified,  
determining the dependent classes of the  
target class.

15. The computer-readable storage medium  
of claim 11, wherein the method further  
comprises:

- a. inserting the object into an object  
database;
- b. determining if the target class and  
supporting classes for the target class are in  
the class path; and
- c. adding the target class and supporting  
classes for the target class to the class path  
if necessary.

For the feature of claim 11 see claim 11  
rejection. For the rest of the claim 15  
feature see claim 6 rejection.

16. The computer-readable storage medium  
of claim 11, wherein the method further  
comprises:

- a. retrieving the object from the object  
database;
- b. determining if the target class and  
supporting classes for the target class are in  
the class path; and
- c. adding the target class and supporting  
classes for the target class to the class path  
if necessary.

For the feature of claim 11 see claim 11  
rejection. For the rest of the claim 16  
feature see claim 7 rejection.

17. The computer-readable storage medium  
of claim 10, wherein the method further  
comprises filtering the list of identified  
classes to remove duplicate and core class  
references.

For the feature of claim 10 see claim 10  
rejection. For the rest of the claim 17  
feature see claim 8 rejection.

20. The apparatus of claim 19, wherein the  
receiving mechanism is additionally  
configured to receive a representation of an  
object;

- a. a serializing mechanism is configured  
to serialize the referenced object;

For the feature of claim 19 see claim 19  
rejection. For the rest of the claim 20  
feature see claim 2 rejection.

- b. a parsing mechanism configured to parse the data resulting from the object serialization to identify classes referenced from the target object's properties, configuration, or state; and
- c. a supporting mechanism that is configured to determine the dependent classes of the target class.

24. The apparatus of claim 20, further comprising:

- a. an insertion mechanism configured to insert the object into an object database;
- b. a determining mechanism configured to determine if the target class and supporting classes for the target class are in the class path; and
- c. an adding mechanism configured to add the target class and supporting classes for the target class to the class path if necessary.

For the feature of claim 20 see claim 20 rejection. For the rest of the claim 24 feature see claim 6 rejection.

25. The apparatus of claim 20, further comprising:

- a. a retrieving mechanism configured to retrieve the object from an object database;
- b. a determining mechanism configured to determine if the target class and supporting classes for the target class are in the class path; and
- c. an adding mechanism configured to add the target class and supporting classes for the target class to the class path if necessary.

For the feature of claim 20 see claim 20 rejection. For the rest of the claim 25 feature see claim 7 rejection.

26. The apparatus of claim 19, further comprising a filtering mechanism configured to filter the list of identified classes to remove duplicate and core class references.

For the feature of claim 19 see claim 19 rejection. For the rest of the claim 26 feature see claim 8 rejection.

### Conclusion

The following summarizes the status of the claims:

35 USC § 102 rejection: Claims 1, 3-5, 10, 12-14, 19, 21-23

35 USC § 103 rejection: Claims 2, 6-8, 11, 15-17, 20, 24-26

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow  
Examiner  
Art Unit 2192  
November 8, 2005

CC



TUAN DAM  
SUPERVISORY PATENT EXAMINER